Graph sketching-based Massive Data Clustering
Anne Morvan12, Krzysztof Choromanski3, Cédric Gouy-Pailler1 and Jamal Atif2
1CEA, LIST, 2Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243, LAMSADE, 3Google Brain Robotics

Objective
We present a new clustering algorithm DBMSTClu providing a solution to the following issues: 1) detecting arbitrary-shaped data clusters, 2) with no parameter, 3) in a space-efficient manner by working on a limited number of linear measurements, a sketched version of the dissimilarity graph G.

Steps of the method
1. The dissimilarity graph of data G with N nodes is handled as a stream of edge weight updates.
2. G is sketched in the dynamic semi-streaming model in one pass over the data into a compact structure requiring O(N polylog(N)) space with method from work [1] relying on ε-sampling principle [2].
3. From the sketch, an AMST is recovered containing N − 1 edges with weight 0 < wj ≤ 1 for all i = 1, . . . , N − 1
4. Apply DBMSTClu on the given AMST (good for expressing the underlying structure of a graph) which successfully detects the right number of non-convex clusters. DBMST is a divisive top-down procedure: at each iteration, a cut among the edges of T is performed creating a new connected component. The best cut to do is identified thanks to a criterion based on Dispersion and Separation of one cluster.

Cluster Dispersion
The Dispersion of a cluster Ci (DISP) represented by the subtree Si of MST T is defined as the maximum edge weight of Si:

\[ \forall i \in [K], \quad \text{DISP}(C_i) = \max_{j, e \in S_i} w_j \]  \quad \text{if} \quad |E(S_i)| \neq 0 \]

\[ \text{otherwise.} \]

(1)

Cluster Separation
The Separation of a cluster Ci (SEP) is defined as the minimum distance between the nodes of Ci and the ones of all other clusters Cj, i ≠ j, 1 ≤ i, j ≤ K, K ≠ 1 where K is the total number of clusters and Cuts(C) denotes the edges incident to cluster Ci:

\[ \forall i \in [K], \quad \text{SEP}(C_i) = \min_{j, e \in \text{Cuts}(C_i)} w_j \]  \quad \text{if} \quad K \neq 1 \]

\[ \text{otherwise.} \]

(2)

Validity Index of a Cluster
The Validity Index of a cluster Ci, 1 ≤ i ≤ K is defined as:

\[ V(C_i) = \frac{\text{SEP}(C_i) - \text{DISP}(C_i)}{\max(\text{SEP}(C_i), \text{DISP}(C_i))} \]

(3)

Validity Index of a Clustering Partition
The Density-Based Validity Index of a Clustering partition P = \{C_i\}, 1 ≤ i ≤ K, DBCVI(P) is defined as the weighted average of the Validity Indices of all clusters in the partition.

\[ \text{DBCVI}(P) = \frac{1}{k} \sum_{i=1}^{k} V(C_i) \]

(4)

Algorithm 1 DBMSTClu algorithm
1. Input: T, the (A)MST
2. splitDBCVI ← −0.1; cut_candidate_list ← \{ edges(T) \}; clusters = [ ]
3. while splitDBCVI < 1.0 do
4. temp_cut ← None; temp_DBCVI ← splitDBCVI
5. for each cut in cut_candidate_list do
6. newClusters ← performCut(clusters, cut)
7. newDVCVI←getDVCVI(newClusters, T)
8. if newDVCVI ≥ temp_DVCVI then
9. temp_cut ← cut; temp_DBCVI ← newDVCVI
10. if temp_cut ≠ None then
11. clusters ← performCut(clusters, temp_cut)
12. splitDBCVI ← temp_DVCVI
13. remove(cut_candidate_list, temp_cut)
14. else
15. break
16. return clusters, splitDBCVI

Conclusion and perspectives
• We introduced a novel non-parametric space-efficient density-based clustering algorithm relying on a sketch built dynamically on the fly as new edge weight updates are received. Its robustness has been assessed by using as input a sketch of the MST rather than the MST itself.
• Further work would be to use DBMSTClu in privacy issues and adapt both the MST recovery and DBMSTClu to the fully online setting i.e. update current MST and clustering partition as new edge weight updates are seen.

References